MP 5 – Illinois James and the Castle of Doom

CS 440 – Fall 2006

Revision $Rev : 27$

Assigned November 27, 2006
Due December 6, 2006
Extension

1 Objectives and Background

The purpose of this MP is to provide the student with some experience in logic programming. Upon completion the student will be familiar with techniques used to search for solutions in Prolog.

You are an explorer who has been locked in a strange castle. The rooms are connected by passages, and the passages are all one-way. You need to find the shortest path from the entry way to the exit. Figure 1 is a map. The numbers show the length of time it takes to go from one room to the other.

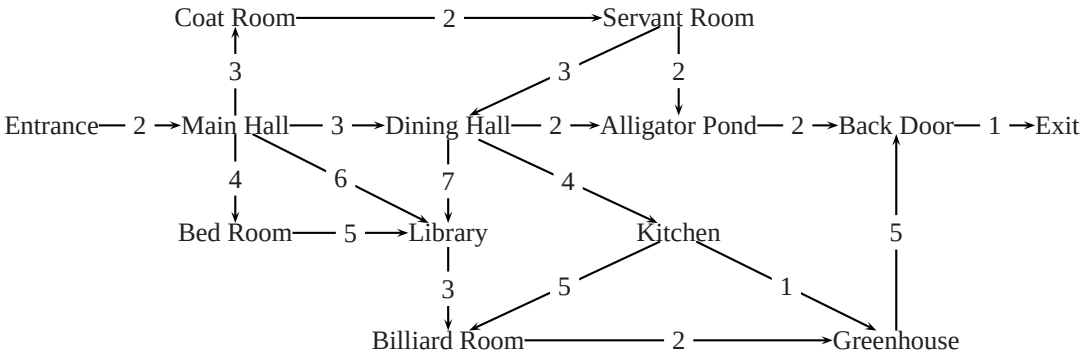

Figure 1: Map of the Castle

There are two other constraints you need to handle. In order to get out the back door, you need a key. Fortunately, there is a key in the servant's quarters, and another in the billiard room. Also, whatever you do, you cannot go thru the alligator pond (I told you this was a strange castle!), because you are allergic to alligators. Besides, even if you weren't, they'd eat you.

2 Overview

Since this is an AI problem, and Prolog is an AI language, you decide to use Prolog to find the path for you.

The file `given.pl`, which is on the web, has the following contents:

```
connected(entrance,mainhall,2).
connected(mainhall,coatroom,3).
connected(mainhall,diningroom,3).
connected(mainhall,library,6).
connected(mainhall,bedroom,4).
connected(bedroom,library,5).
connected(coatroom,servantroom,2).
connected(diningroom,alligatorpond,2).
connected(diningroom,kitchen,4).
connected(diningroom,library,7).
connected(servantroom,alligatorpond,2).
connected(servantroom,diningroom,3).
connected(alligatorpond,backdoor,2).
connected(backdoor,exit,1).
connected(library,billiardroom,3).
connected(kitchen,billiardroom,5).
connected(kitchen,greenhouse,1).
connected(billiardroom,greenhouse,2).
connected(greenhouse,backdoor,5).
```

Basically, it is an encoding of the map of the castle. To use the file, start up Prolog and type `[given].` to ”use” the file.

Here is a sample session on host220:

```
% pl
Welcome to SWI-Prolog (Multi-threaded, Version 5.4.7)
Copyright (c) 1990-2003 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- [given].
% given compiled 0.01 sec, 5,784 bytes

Yes
?- connected(diningroom,X,C).

X = alligatorpond
C = 2 ;

X = library
C = 7 ;

X = kitchen
C = 4 ;

No
?-
```

Here we have asked Prolog which rooms we can get to from the diningroom, and how long it takes.

For your own MP, copy `given.pl` to your own file `mp7.pl` and edit that. **You may not change, add to, or delete any of the connected predicates.**

3 Problems

1. The first thing we need is a way to see if there is a path between two rooms. Make a predicate called `pathfrom(X,Y,L)` which is true whenever L is a list of rooms that form a valid path from X to Y. Example:

```
?- pathfrom(mainhall,billiardroom,P).
P = [mainhall, bedroom, library, billiardroom] ;
P = [mainhall, coatroom, servantroom, diningroom, kitchen, billiardroom] ;
P = [mainhall, coatroom, servantroom, diningroom, library, billiardroom] ;
P = [mainhall, diningroom, kitchen, billiardroom] ;
P = [mainhall, diningroom, library, billiardroom] ;
P = [mainhall, library, billiardroom] ;
No

?- pathfrom(coatroom,library,P).
P = [coatroom, servantroom, diningroom, library] ;
No

?- pathfrom(X,kitchen,P).

X = diningroom
P = [diningroom, kitchen] ;

X = entrance
P = [entrance, mainhall, coatroom, servantroom, diningroom, kitchen] ;

X = entrance
P = [entrance, mainhall, diningroom, kitchen] ;

X = mainhall
P = [mainhall, coatroom, servantroom, diningroom, kitchen] ;

X = coatroom
P = [coatroom, servantroom, diningroom, kitchen] ;

X = mainhall
P = [mainhall, diningroom, kitchen] ;

X = servantroom
P = [servantroom, diningroom, kitchen] ;

No

?-
```

2. Now you want to find out how long it will take to go from one room to the other. Write a predicate `costof(L,C)`, which is true when C is the cost of taking the path specified by the list L.

```
?- costof([mainhall,diningroom,kitchen],C).

C = 7 ;
```

```
No

?-
```

3. As you have seen, there can be multiple paths from one room to another. Write a predicate `costoflist(P,C)` that is true when C is a list of costs corresponding to the list of paths P.

```
?- costoflist([[mainhall, bedroom,library,billiardroom],[mainhall,library,billiardroom]]

C = [12, 9] ;

No
?-
```

4. Now we are almost ready to find out which path is shortest thru the castle. We need a predicate `minlist(X,L)` which is true when X is the minimum element of L. Hint: it might be easier to write a helper predicate `leqlist(X,L)` which is true if X is less than or equal to every element in L. You don't have to do this, though. If you do, be aware that Prolog uses =< instead of <=.

```
?- minlist(2,[3,1,2,5]).

No
?- minlist(1,[3,1,2,5]).

Yes

?- minlist(M,[5,3,8,2,4]).

M = 2 ;
No
?-
```

5. At this point you have a list of paths, and a list of how much it costs to take each path. Write a predicate `mincost(P,C,PL,CL)` which is true if P and C hold the same position in their respective lists, and C is the minimum element of the list CL.

```
?- mincost(a,4,[s,a,q],[6,4,8]).

Yes
?- mincost(P,C,[s,a,q],[6,4,8]).

P = a
C = 4 ;

No
?- mincost(P,C,[[mainhall, bedroom,library,billiardroom],[mainhall,library,billiardroom

P = [mainhall, library, billiardroom]
```

```
C = 9 ;

No
?-
```

Hint: the following predicate may be useful:

```
member2(A,B,[A|_],[B|_]).
member2(A,B,[_|X],[_|Y]) :- member2(A,B,X,Y).
```

6. Now you are ready to write the search routine. Write a predicate `findpath(A,B,P,C)` which is true when P is the lowest cost path between A and B, with cost C. Hint: try typing in `findall(X,pathfrom(mainhall,billiardroom,X)` and see what is put into L.

```
?- findpath(entrance,exit,Path,Cost).

Path = [entrance, mainhall, diningroom, alligatorpond, backdoor, exit]
Cost = 10 ;

No
?-
```

There! You have the shortest path between the entrance and the exit. Unfortunately, that path doesn't go near a room with a key, and further, that path also goes thru the alligator pond.

7. Write a new predicate `goodpath(L)` which is true only if L does not contain the alligator pond, *and* goes thru either the servant quarters or the billiard room.

```
?- goodpath([entrance, mainhall, diningroom, alligatorpond, backdoor, exit]).

No
?- goodpath([entrance, mainhall, diningroom, backdoor, exit]).

No
?- goodpath([entrance, mainhall, diningroom, servantroom, backdoor, exit]).

Yes
```

Note that `goodpath` only checks to see if there is are certain rooms in the path, it doesn't need to verify that it is a real path.

8. Last predicate! Write a predicate `findgoodpath` which is like `findpath`, but returns the minimum size path that avoids the alligator room and goes thru one of the rooms with a key.

```
?- findgoodpath(entrance,exit,Path,Cost).

Path = [entrance, mainhall, library, billiardroom, greenhouse, backdoor, exit]
Cost = 19 ;

No
?-
```

Hint: this might be easier if you first write a predicate `goodpathfrom`.

4 Getting Started and Handing In

There is a Prolog tutorial at http://www.csupomona.edu/ j̃rfisher/www/prolog_tutorial/contents.html.

Also, the SWI homepage is at http://www.swi-prolog.org/. There you will find versions you can install on your own computer, and instruction manuals.

Handing in will be via subversion. Please create a `mp5` subdirectory and name your file `netid.pl`, of course substituting your own email id for `netid`.